

Wer heute effizient Apps entwickeln will, die auf möglichst vielen Geräten und Gerätetypen laufen, muss plattformübergreifend arbeiten. Dafür gibt es bereits eine Reihe zuverlässiger Frameworks.

Von Timo Kockert*



Konzepte für die mobile

Eine Zeitlang sah es so aus, als würde sich das Thema Cross-Platform-Entwicklung für die IT erledigen. Die Vielfalt der proprietären Plattformen, Systeme und Systemvarianten, die zu Anfang der 80er Jahre die IT beherrschte und die eine Anwendungsentwicklung auf breiter Basis oft lähmte, konsolidierte sich nach und nach. Schließlich blieben bei Großrechnern nur eine Handvoll und bei verteilten Systemen im Wesentlichen nur noch Windows und Linux übrig sowie Mac OS als Nische, für die immer schon andere Gesetze galten.

Mit dem Übergang zu einer mehrschichtigen Anwendungsarchitektur, zunächst auf Basis von komponentenorientierten Ansätzen wie Corba, später dann verstärkt mit Web-Services, wurde eine plattformübergreifende Softwareentwicklung möglich, mit der sich breite Märkte erschließen ließen.

Konsolidierung schreitet voran

Mittlerweile hat sich das Bild gewandelt. Der Konsolidierung im Bereich Server und Desktop-Rechner steht eine andere Tendenz entgegen: Mobile Systeme wie Smartphones und Tablet-PCs haben nicht nur eine Vielzahl neuer Anwendungen gebracht, sondern auch eine fast unüberschaubare Menge an unterschiedlichen technischen Konzepten, Plattformen und Gerätetypen, die untereinander meist nur wenig kompatibel sind. Grundsätzlich kann man sich bei der Entwicklung auf die fünf großen Systeme iPhone, Android, Windows Phone, BlackBerry und Symbian konzentrieren. iPhone-Apps werden von Unternehmen derzeit zwar am meisten nachgefragt, die größte

Verbreitung hat aber Android, während Symbian-Geräte nach wie vor eher als Telefon benutzt werden.

Viel Aufwand für Entwickler

Auch auf den einzelnen Plattformen gibt es zum Teil größere gerätetypische Unterschiede mit unterschiedlichen Bildschirmgrößen und -auflösungen, vor allem bei Android, weil dafür viele Hersteller Geräte auf den Markt bringen. Die daraus entstehende Aufgabe sollten Softwareentwickler nicht unterschätzen. Adesso Mobile Solutions zum Beispiel hat im eigenen Haus eine Multi-Channel-Plattform aufgebaut, deren

Gute Frameworks stehen Open Source zur Verfügung.

Datenbank mittlerweile rund 18.000 unterschiedliche Endgeräteprofile und spezifische Funktionen umfasst.

Unternehmen, die planen, Software für Smartphones und Tablets bereitzustellen, wollen und können sich in der Regel nicht mit einer rein nativen Softwareentwicklung auf eine bestimmte Plattform festlegen. Sie müssen ihre Lösungen so konzipieren, dass sie einen möglichst breiten Markt abdecken, umso mehr, als man nicht bei jeder neuen Gerätegeneration mit der Entwicklungsarbeit von vorne beginnen will. Daher spielt gerade für diese Systeme die Cross-Platform-Entwicklung eine so wichtige Rolle.

Grundsätzlich lässt sich auch bei der Entwicklung von Cross-Platform-Applikationen für mobile Systeme zwischen zwei Ebenen unterscheiden: Es gibt zum einen plattform-spezifische Funktionen und zum anderen Kernprozesse, die für alle Systeme gleich sind und aus wiederverwendbaren Komponenten bestehen. Die spezifischen Funktionen betreffen vor allem die unterschiedlichen Benutzeroberflächen, aber auch – und das macht die Entwicklung hier so komplex – eine Reihe spezifischer Features wie Bewegungssensor, GPS-Modul, Adressbuch, Soundeffekte oder Kameras.

Diese Features gehören zu den typischen Leistungen eines mobilen Geräts, und kein Benutzer wird bei seinem Smartphone ausgerechnet darauf verzichten wollen. So werden von etlichen Unternehmen beispielsweise standortbezogene Dienste entwickelt, die genau auf derartigen Features aufsetzen, und in vielen Fällen gehören diese Dinge zu den wichtigsten Motiven, überhaupt eine App bereitzustellen.

Ein weiterer Aspekt ist zu berücksichtigen: So unterschiedlich die Geräte in ihren technischen Details auch sein mögen, die Benutzer erwarten heute, dass sie alle einigermaßen gleich funktionieren und aussehen, also eine gewisse Einheitlichkeit aufweisen.

Kein Mangel an Frameworks

Für eine effiziente Softwareentwicklung, die nicht eine komplette App für jede Plattform erstellen kann, sondern gemeinsame Basisfunktionen mit plattform-spezifischen verbindet, steht heute eine Reihe von Cross-Platform-Frameworks zur Verfügung. ▶



Cross-Platform-Entwicklung

► Das Angebot wächst auch hier ständig, aber Lösungen wie QuickConnectFamily, iPFaces mobile, MotherApp, MoSync, Tersus oder Worklight haben im Markt (noch) keine relevante Position.

Mehr Bedeutung haben da schon PhoneGAP, Titanium und auch Rhodes. Alle drei stehen als Open-Source-Werkzeuge zur Verfügung, so dass Entwickler ihr Framework nach Bedarf anpassen können. Das wird wichtig, wenn im Unternehmen bereits andere Frameworks im Einsatz sind.

✗ **PhoneGAP** ist ein Framework, das für die Oberflächenentwicklung einer App ein normales Web-Seiten-Design mit HTML und CSS unterstützt. Die nativen Funktionen, beispielsweise der Zugriff auf das Adressbuch oder den Bewegungssensor, die man nicht durch das Web-Seiten-Design abdecken kann, stellt PhoneGAP über eine plattformübergreifende Javascript-API bereit. Das Framework unterstützt die Plattformen iPhone, Windows Mobile, BlackBerry, Symbian, Palm und Android, also eine recht breite Palette. Mit PhoneGAP lassen sich Apps erstellen, die deutlich mehr können als eine reine Web-Seite. Sie benötigen zur Ausführung keinen Browser, und man kann sie wie jede richtige App auch in die betreffenden App-Stores einstellen. Allerdings ist die Benutzerschnittstelle ganz in HTML designt, was man an einigen Stellen durchaus sieht und spürt. Anders ausgedrückt: Eine PhoneGAP-App ist am Ende doch keine hundertprozentige App. Für Optimierungen des typischen Look-and-Feels stehen zusätzliche Libraries zur Verfügung, aber der Unterschied zu einer nativen App bleibt doch erkennbar.

✗ **Appcelerator Titanium** ist ein Framework, bei dem die gesamte App in Javascript geschrieben wird, also nicht nur die Anwendungslogik, sondern auch die grafische Benutzeroberfläche. Aus dem Javascript-Code generiert Titanium nativen Code für die unterschiedlichen Plattformen. Die Benutzerelemente wie Buttons werden daher nativ erstellt, was einer App ein ganz anderes Look-and-Feel verleiht. Die Elemente arbeiten sehr flüssig zusammen, so dass eine Titanium-Applikation für den Benutzer nicht mehr von einer nativ programmierten App zu unterscheiden ist. Der Nachteil von Titanium ist, dass zurzeit mit iPhone und Android nur zwei Plattformen unterstützt



Ausgewählte Frameworks

PhoneGap

Funktionsweise: Oberflächen werden mit HTML und CSS realisiert wie bei gewöhnlichen Websites; Applikationslogik und spezielle Gerätefunktionen können über ein plattformübergreifendes Javascript aufgerufen werden.



➤ Existierende HTML, CSS, Javascript-Skills können verwendet werden, ebenso vorhandene Javascript-Frameworks.

➤ Oberflächen nicht nativ, dürftige Dokumentation.

Tutorial: <http://w.idg.de/rJB379>

Rhodes

Funktionsweise: Oberflächen werden mit HTML und CSS realisiert, Applikationslogik und Gerätefunktionen in Ruby implementiert.



➤ Gute Trennung von Design und Logik nach dem Model-View-Controller-Ansatz.

➤ Oberflächen nicht nativ.

Tutorial: <http://w.idg.de/v9cCla>

Appcelerator Titanium

Funktionsweise: Oberflächen und Applikationslogik werden mit Javascript implementiert.



➤ Entwicklung nativer Oberflächen.

➤ Geringe Plattforunterstützung.

Tutorial: <http://w.idg.de/tycZA3>

werden. Die Unterstützung von BlackBerry ist in Vorbereitung, und auf Dauer wird wohl auch Windows Phone folgen.

✗ **Rhodes** verfolgt einen ähnlichen Ansatz wie PhoneGAP. Auch hier wird die Oberfläche mit HTML und CSS erstellt – sie ist also nicht nativ. Die Interaktion zwischen den Web-Seiten läuft bei Titanium über die Programmiersprache Ruby, in der auch die Applikationslogik implementiert wird. Design und Logik werden bei Rhodes strikt getrennt. Rhodes ist auf die Erstellung von Business-Anwendungen ausgerichtet und bietet eine einfache Persistenzlösung inklusive Synchronisation mit dem Server. Unterstützt werden iPhone, Android, Windows Mobile, BlackBerry und Symbian.

Wiederverwendbare Programme

Mit diesen Frameworks lässt sich die Cross-Platform-Entwicklung erheblich vereinfachen; man kann auch auf native Programmierung verzichten und in hohem Maße wiederverwendbare Programme erstellen. Allerdings geht der Cross-Platform-Ansatz in der Praxis nicht ganz so weit, dass man eine App nur ein einziges Mal programmieren muss und

die plattformspezifischen Varianten dann beim Neukompilieren ganz automatisch erzeugt werden. So sind beispielsweise bei PhoneGAP einige gerätespezifische Anpassungen erforderlich, weil HTML für das iPhone anders gerendert wird als für Android. Damit die Apps wirklich gleich aussehen, müssen daher noch Detailanpassungen vorgenommen werden. Dabei sollte man die Unterschiede der Geräte hinsichtlich ihrer technischen Ausstattung – beispielsweise bei den verfügbaren Bedienelementen – berücksichtigen, so dass ein identisches Aussehen von Apps auch bei nativer Programmierung nicht möglich ist.

Die Nachfrage nach Apps wird auf absehbare Zeit weiter zunehmen. Mobile Anwendungen mit Smartphones oder Tablets gehören zu den großen Trends in der IT, und ihr Einsatz wird sich auch im Business-Bereich weiter intensivieren. Dafür wird eine entsprechende Anzahl von Apps benötigt, die natürlich auch Business-Prozesse abdecken müssen. Es wird künftig jedenfalls nicht mehr ausreichen, einfach irgendeine App bereitzustellen, nur um mit „dabei“ zu sein. Die Anforderungen an die Softwareentwicklung werden steigen. (hv)

*Timo Kockert ist Software Engineer bei der Adesso Mobile Solutions GmbH in Dortmund.